CVS
Concurrent Versions System
Manual de Usuario

Resumen

Sistema de control de versiones basado en código *open-source* que mantiene el registro de todo el trabajo y los cambios en los archivos (código fuente principalmente) que forman un proyecto y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren.

Índice

1. Introducción

Cvs es un sistema de control de versiones con el que se puede mantener un historial del código. Se puede acceder a cualquier version anterior ya que CVS guarda en un sólo archivo las diferencias de todas las versiones logrando así utilizar mucho menos espacio en disco.

Cvs tambien es de gran ayuda cuando se trabaja con un grupo de personas en un mismo proyecto. Cada desarrollador trabaja en su propia copia del directorio y CVS fusiona el trabajo cuando cada desarrollador terminó, resolviendo los posibles conflictos.

Con CVS se puede trabajar de forma local (repositorio y copias de trabajo en el mismo sistema) o remota (el repositorio está en un sistema servidor y la copia local en otro que es cliente del primero).

2. Algunas Definiciones

Repositorio: Jerarquía de directorios alojada en el servidor CVS que contiene diferentes módulos a disposición de los usuarios.

Módulo: Árbol de directorios que forma parte del repositorio. Cuenta con un nombre identificador gracias al cual podremos trabajar con él de forma selectiva.

3. Configuración

Se pueden usar varios archivos de configuración que CVS reconocerá y usará.
/.cvsignore que contiene los sufijos de los archivos que no nos interesa que CVS controle.
/.cvsrc que contiene aquellos parámetros que CVS usará cada vez que se invoque una determinada orden, de forma automática.

4. Autenticación

Al trabajar en remoto con CVS pueden elegirse varias alternativas de autenticación: Las más utilizadas son vía pserver y vía ssh.

4.1. ssh (OpenSSH)

Para que CVS use este modo de autenticación se deben usar estas variables de entorno:

```
export CVSROOT=:ext:USUARIO@cvs.dominio.org:/var/lib/cvs
export CVS_RSH=/usr/bin/ssh
```

donde USUARIO es el nombre de usuario que tiene acceso al repositorio, cvs.dominio.org, es el nombre del servidor donde se aloja el

repositorio, /var/lib/cvs es el directorio del servidor en el que está el repositorio y /usr/bin/ssh es la ruta completa al ejecutable de ssh.

Hay que tener en cuenta que usando esta técnica, tendrá que autenticarse (es decir, suministrar su contraseña) cada vez que ejecute alguna orden de CVS (a menos que use autenticación con clave pública RSA/DSA para ssh). La ventaja de usar ssh como método de autenticación es que las comunicaciones con el servidor CVS van completamente cifradas, tanto la autenticación como los datos que intercambiemos con el servidor (cosa que no ocurre con el siguiente método). El inconveniente de este método de autenticación es que deberá crear cuentas de usuarios locales en el servidor CVS con posibilidad de inicio de sesión (shell válido) para todos aquellos usuarios remotos que necesiten acceso al servidor, lo cual implica un acceso más amplio al equipo donde se ejecuta el servidor CVS que el mero acceso al servicio CVS.

5. Modo de Uso

Antes de cada sesión de trabajo es conveniente hacer

```
cvs update -Pd
```

para asegurarnos de que disponemos de las últimas modificaciones registradas en el repositorio. Justo al acabar cada sesión de trabajo es conveniente hacer cvs commit (se puede abreviar en cvs ci) para que todas nuestras modificaciones se registren en el repositorio.

5.1. Descarga del módulo por primera vez

Para crear una copia de trabajo local del módulo CVS deseado debemos usar el comando cvs checkout(abreviable como cvs co):

```
$ cd padre-de-directorio-donde-se-alojará-el-módulo
$ cvs checkout nombre-del-módulo
```

Esto creará una jerarquía de directorios donde se almacenará la copia local de trabajo el módulo. Este paso sólo hay que hacerlo una vez por cada módulo. A partir de este momento no es necesario configurar las variables de entorno porque CVS sabe a qué repositorio pertenece el módulo con sólo examinar los subdirectorios CVS. No se debe modificar nunca esos subdirectorios a mano. De lo contrario CVS perderá la pista de a que módulo pertenecen los archivos, cuáles son las versiones de la copia local, etc.

5.2. Actualizar nuestra copia local desde el repositorio

Cuando queremos actualizar la copia local de trabajo del módulo con los cambios que hayan podido hacer otros usuarios y que están recogidos en el repositorio deberemos hacer:

```
$ cd directorio-del-módulo
$ cvs update -Pd
```

5.3. Publicar nuestras modificaciones en el repositorio

Se usa el comando *commit (o su equivalente ci)*:

```
$ cd directorio-del-módulo
$ cvs commit
```

Tras lo cual el sistema mostrará la pantalla de un editor de textos (el que tengamos configurado como nuestro favorito en la variable de entorno EDITOR) para que introduzcamos una descripción, lo más significativa posible, del conjunto de cambios realizados en el módulo desde el último commit.

6. Resolución de conflictos

Habrá ocasiones en las que tengamos que resolver los conflictos que surjan entre diferentes versiones de un módulo o archivo del mismo (recuerde que puede haber múltiples personas trabajando de forma concurrente sobre el mismo módulo) para que CVS continúe trabajando. Estos conflictos son normales y ocurren cuando dos o más personas modifican a la vez exactamente la mismas partes de un archivo.

El procedimiento es simple:

- CVS se quejará de un archivo al hacer un update o un commit.
- Editamos ese archivo y encontraremos marcas del tipo:

```
[...]
>>>>>>>>>>>>>>
texto-opción-1
===========
texto-opción-2
<<<<<<<<<<<<<<
[...]
```

- El texto entre marcas es el que produce el conflicto. Hay que elegir qué modificación nos gusta y borramos todo lo demás.
- Si no quedan más conflictos volvemos a hacer el commit o update.

7. Añadir archivos al módulo

No olvide que CVS controlará sólo los archivos que se hayan descargado inicialmente desde el repositorio. Cualquier otro archivo o directorio de la jerarquía del módulo CVS será ignorado. Si quiere añadir un nuevo archivo o directorio al módulo CVS hay que seguir los siguientes pasos (ademas de crear o copiar el propio archivo al módulo, por supuesto):

```
$ cd directorio-del-módulo
$ cvs add archivo
```

pero si el archivo es binario hay que tener la precaución de hacer:

```
$ cd directorio-del-módulo
$ cvs add -kb archivo
```

CVS usa varias variables (en realidad son de RCS, que funciona por debajo de CVS). Si el archivo es binario es posible que se dé una combinación de bytes que coincidan con alguna de estas variables. Si así fuera, RCS/CVS modificaría el contenido y lo corrompería. También se debe a que el sistema de cálculo de diferencias que usan estos sistemas no está diseñado para trabajar con información binaria. Si se obra equivocadamente es probable que corrompamos los datos.

Si bien se pueden gestionar archivos binarios, no se hará control de versiones de los mismos. Sólo se guardará la última versión (al no disponer CVS de la funcionalidad necesaria para calcular diferencias de archivos binarios). Tras la orden cvs add hay que hacer ejecutar de nuevo el comando cvs commit para incluir los nuevos archivos en el repositorio CVS.

8. Interfaces de usuario para CVS

- **pharmacy:** Una interfaz GNOME para CVS, disponible en http://pharmacy.sourceforge.net/. Se encuentra aun en un estado de desarrollo bastante temprano y no se actualiza
- **cvsgui:** Una interfaz multiplataforma para CVS escrita en C++ (anteriormente conocida como gCVS), disponible en http://sourceforge.net/projects/cvsgui/.
- **tkcvs:** Interfaz gráfica para CVS escrita en Tcl/Tk muy establecida y estable, disponible en http://www.twobarleycorns.net/tkcvs.html.
- **cervisia:** Interfaz gráfica KDE para CVS, disponible en http://cervisia.kde.org/.
- **PCL-CVS:** extensión de (X)Emacs que permite manipular archivos gestionados con CVS de forma automática y transparente, disponible como parte de XEmacs, y como parte del propio CVS.